자료구조 6주차 과제

충북대학교 컴퓨터공학과 2022040014 오재식

Matrix.c

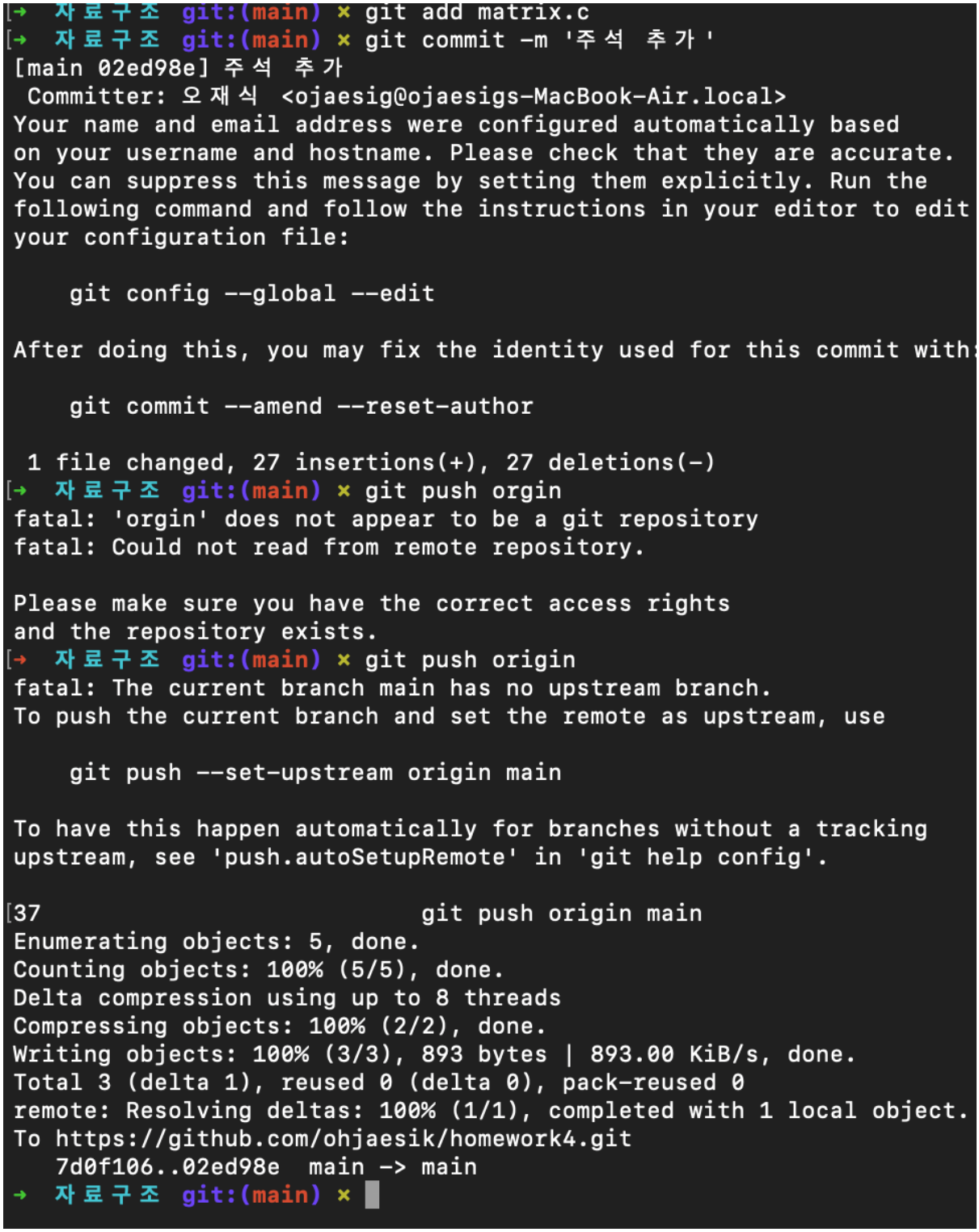
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

void print_martix(int** matrix, int row, int column){
    printf("matrix \n");
    for(int i = 0; i < row ; i++){ //1 행부터 row 행 까지 출력
        for(int j = 0; j < column; j++){ //1 열부터 column 열까지 출력
            printf("%d ", matrix[i][j] );
        }
        printf("\n");
    }
    printf("\n");
}
void free_matrix(int ** matrix, int row){

        for(int i=0; i<row; i++)
            free(matrix[i]); //할당된 메모리 해제
        free(matrix); //할당된 메모리 해제
}

void addition_matrix(int **A, int **B, int A_n, int A_m, int B_n, int B_m){
    if(A_n != B_n || A_m != B_m){
        printf("A and B cannot be added \n"); //더할 수 없는 행렬인 경우 출력

    }
    else{
        int ** added_matrix = (int **)malloc(sizeof(int *) * A_n); //A+B 의 값을
저장하기 위한 새로운 배열을 만듬

        for (int i =0; i < A_n; i++){
            added_matrix[i] = (int *)malloc(sizeof(int) * A_m); //A+B 의 값을 저장하기
위한 새로운 배열을 만듬
            for(int j =0; j< A_m; j++){
                added_matrix[i][j] = A[i][j] + B[i][j]; //A 의 i 행 j 열과 B 의 i 행
j 열을 더하고 그 값을 added_matrix 에 할당.
            }
        }
        print_martix(added_matrix, A_n, A_m);
        free_matrix(added_matrix, A_n);
    }
}

void subtraction_matrix(int **A, int **B, int A_n, int A_m, int B_n, int B_m){
    if(A_n != B_n || A_m != B_m){
        printf("A and B cannot be subtracted \n"); //뺄 수 없는 행렬인 경우 출력
```

```
    }
    else{
        int ** subtracted_matrix = (int **)malloc(sizeof(int *) * A_n); //A-B 의
값을 저장하기 위한 새로운 배열을 만듬.

        for (int i =0; i < A_n; i++){
            subtracted_matrix[i] = (int *)malloc(sizeof(int) * A_m); //A-B 의 값을
저장하기 위한 새로운 배열을 만듬.
            for(int j =0; j< A_m; j++){
                subtracted_matrix[i][j] = A[i][j] - B[i][j]; //A-B 의 행렬 값을
subtracted_matrix 에 저장.
            }
        }
        print_martix(subtracted_matrix, A_n, A_m);
        free_matrix(subtracted_matrix, A_n);
    }
}

void transpose_matrix(int **matrix,int row,int column){
    int ** transpose_matrix = (int**)malloc(sizeof(int*) * column); //전치 행렬을
만들기 위한 배열을 만듬
    for(int i =0; i < column; i++){
        transpose_matrix[i] = (int*)malloc(sizeof(int) * row); //전치행렬을 만들기
위한 배열을 만듬
    }
    for(int i = 0;i < row; i++){
        for(int j=0; j < column; j++){
            transpose_matrix[j][i] = matrix[i][j]; //matrix 의 i 행 j 열을
transpose_matrix 의 j 행 i 열에 할당
        }
    }
    print_martix(transpose_matrix, column, row);
    free_matrix(transpose_matrix, column);
}

void multiply_matrix(int **A, int **B, int A_n, int A_m, int B_n, int B_m){
    if(A_m != B_n){
        printf("A cannot be multiplied by B \n"); //행렬의 곱이 불가한 경우 출력

    }
    int **multiply_matrix = (int **)malloc(sizeof(int*) * A_n); //행렬의 곱의 결과를
저장할 배열 생성
    for(int i =0; i < A_n; i++){
        multiply_matrix[i] = (int *)malloc(sizeof(int) * B_m); //행렬의 곱의 결과를
저장할 배열 생성
        for(int j=0; j< B_m; j++){
```

```
            multiply_matrix[i][j] = 0; //행렬의 곱을 저장할 배열을 초기화해줌.
            for(int k=0; k < B_n; k++){
                multiply_matrix[i][j] += A[i][k] * B[k][j]; // 행렬의 곱의 결과를
저장함.
            }

        }
    }
    print_martix(multiply_matrix,A_n, B_m);
    free_matrix(multiply_matrix, A_n);
}

int main(){
    int A_n , A_m, B_n, B_m;
    printf("----- ohjaesik 2022040014 ----- \n");
    printf("input A matrix size : ");
    scanf("%d %d", &A_n, &A_m);
    printf("input B matrix size : ");
    scanf("%d %d", &B_n, &B_m);

    int ** A_martix = (int **)malloc(sizeof(int *) * A_n); //행렬 A 의 행
    int ** B_martix = (int **)malloc(sizeof(int *) * B_n); //행렬 B 의 행

    for(int i =0; i <A_n; i++){
        A_martix[i] = (int *)malloc(sizeof(int)* A_m); //행렬 A 의 열
    }
    for(int i=0; i< B_n; i++){

        B_martix[i] = (int *)malloc(sizeof(int)* B_m); //행렬 B 의 열
    }
    for(int i=0; i<A_n; i++){
        for(int j=0; j<A_m; j++){
            A_martix[i][j] = rand() % 10; //행렬 A 의 임의의 값 할당
        }
    }
    for(int i=0; i<B_n; i++){
        for(int j=0; j<B_m; j++){
            B_martix[i][j] = rand() % 10; //행렬 B 의 임의의 값 할당
        }
    }
    printf("--- A_matrix --- \n");
    print_martix(A_martix, A_n, A_m);

    printf("--- B_matrix --- \n");
    print_martix(B_martix,B_n, B_m);

    printf("--- A+B matrix --- \n");
    addition_matrix(A_martix,B_martix,A_n,A_m,B_n,B_m);
```

```c
    printf("--- A-B matrix --- \n");
    subtraction_matrix(A_martix, B_martix, A_n, A_m, B_n, B_m);

    printf("--- Transpose matrix --- \n");
    transpose_matrix(A_martix,A_n,A_m);

    printf("--- A*B matrix --- \n");
    multiply_matrix(A_martix,B_martix,A_n,A_m,B_n,B_m);
    return 0;
}
```

```
----- ohjaesik 2022040014 -----
input A matrix size : 3 3
input B matrix size : 3 2
--- A_matrix ---
matrix
7 9 3
8 0 2
4 8 3

--- B_matrix ---
matrix
9 0
5 2
2 7

--- A+B matrix ---
A and B cannot be added
--- A-B matrix ---
A and B cannot be subtracted
--- Transpose matrix ---
matrix
7 8 4
9 0 8
3 2 3

--- A*B matrix ---
matrix
114 39
76 14
82 37
```



https://github.com/ohjaesik/homework4